UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/799,351	03/12/2004	David Dehghan	MSFT122464	6346

26389 7590 09/25/2007

CHRISTENSEN, O'CONNOR, JOHNSON, KINDNESS, PLLC
1420 FIFTH AVENUE
SUITE 2800
SEATTLE, WA 98101-2347

EXAMINER	
LE, MIRANDA	

ART UNIT	PAPER NUMBER
2167	

MAIL DATE	DELIVERY MODE
09/25/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/799,351	DEHGHAN ET AL.	
	Examiner	Art Unit	
	Miranda Le	2167	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *06 July 2007*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-17 and 25-29* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-17 and 25-29* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *08/23/2004*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This communication is responsive to Amendment, filed 07/06/2007.

Claims 1-17, 25-29 are pending in this application. Claims 1, 3, 4, 11, 25, 26, 26, 28, 29 have been amended, claims 18-24 have been cancelled. This action is made Final.

2. The objection to the specification (drawings, claim objection) of the invention has been withdrawn in view of the amendment.

Information Disclosure Statement

3. The information disclosure statement (IDS) submitted on 08/23/2004 is in compliance with the provisions of 37 CFR 1.97. Accordingly, the information disclosure statement is being considered by the examiner.

Drawings

4. The drawings were received on 07/06/2007. These drawings are acceptable.

Claim Rejections - 35 USC § 102

5. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

6. Claims 11-17 are rejected under 35 U.S.C. 102(b) as being anticipated by Donohue (US Patent No. 6,199,204).

Donohue anticipated independent claim 11 by the following:

As per claim 11, Donohue teaches an update service node for distributing software updates to client computers and to child update service nodes, wherein the update service node is organized in a hierarchy of a plurality of similarly configured update service nodes *(i.e. Thus, the present invention provides an agent and a method for obtaining and applying software updates which significantly reduces the cost and effort for software distributors of distributing and tracking software updates and significantly reduces the effort for system administrators and end users of applying updates to installed software, col. 5, lines 43-48),* the update service node comprising:

an update store for storing software updates *(i.e. repository 40', col. 7, lines 12-20;*

an update web service through which the update service node obtains software updates from a parent update service node over a communication network, and through which the update service node distributes software updates to child update service nodes in the hierarchy over the communication network *(i.e. repository 40', col. 7, lines 12-20; For an updater component to find and talk to another updater component on another remote machine the above information would have to be augmented by having a repository 40' which is accessible from both machines (preferably a central or distributed database accessible from anywhere in the network, such as a Web Page or pan-network file) and is available to all updater components that require it. Entries would be of the form updater_name machine_ip_address (OR DNS entry), port number, protocol, col. 7, lines 12-20);*

an administration application programming interface (API) through which an administrator *(i.e. registration, col. 7, lines 34-45)*, using an administration user interface, establishes rules for distributing software updates to its child update service nodes *(i.e. The*

structure of an updater component comprises data, methods for operating on that data, and a public application programming interface (API) which allows other updater components to contact and communicate with it. This structure will now be described in detail, col. 11, lines 21-25); and

a child update module for determining which software updates are available to be distributed to its child update service nodes according to the established rules *(i.e. If it finds all the updater components located locally or remotely, it can be sure that the software pre-requisites are available and it next needs to contact each updater component for each software product to be sure all pre-requisites are at the correct level. If an updater component 20' having a required product identifier for pre-requisite software 30' but not having the required version number is found locally or remotely, and if forcing of updates is the update policy, col. 13, lines 22-41).*

As per claim 12, Donohue teaches the update service node of Claim 11, further comprising:

a reporting module for generating and sending update activity reports to the parent update service node *(i.e. The updater generates 380 a report and writes 390 to log records, and then quits execution 400 (in the preferred embodiment the updater goes into a sleep or idle state) until activated again 410 upon expiry of a predetermined update cycle period (the repeat period parameter is configured when the updater component is installed), col. 11, lines 14-19);*

an authentication and authorization module for determining whether an update service mode is authorized to obtain software update from the child update service node *(i.e. The*

updater component preferably also includes a mechanism for verifying the authenticity of downloaded software, using cryptographic algorithms. This avoids the need for dedicated, password-protected or otherwise protected software resource repository sites. The software resources can be anywhere on the network as long as they are correctly named and or posted to the network search engines, col. 5, lines 36-42); and

a client update module for distributing software updates to client computers *(i.e. user, col. 3, line 65 to col. 4, line 14).*

As per claim 13, Donohue teaches the update service node of Claim 12, wherein the update store comprises an update content store in which the update payload for the software update is stored, and an update information store in which update metadata for the software update is stored *(i.e. Product_ID; Current_Installed_Version; Current_License, col. 11, lines 26-39).*

As per claim 14, Donohue teaches the update service node of Claim 13, wherein, wherein the update service node obtains the software update from the parent update service by obtaining update metadata for the software update from the parent update service node, and separately obtaining the update payload for the software update from the parent update service node *(i.e. Since the updates list file 160 returned to the updater component in response to an initial search includes an identification 130 of pre-requisite software, that information enables the aforementioned examination 290 of the updater component registration database 40,40' to check whether pre-requisite software is available locally or remotely, col. 13, lines 22-41).*

As per claim 15, Donohue teaches the update service node of Claim 14, wherein the update service node obtains the update payload for the software update from the parent update service node in just-in-time fashion *(i.e. Each server system includes within storage a list 60 of the latest versions of, and patches for, software products which are available from that server. Each vendor is assumed here to make available via their Web sites such a list 60 of software updates (an example of which is shown in FIG. 2) comprising their product release history, in a format which is readable by updater components, and to make available the software resources 70 required to build the releases from a given level to a new level, col. 7, line 55 to col. 8, line 11).*

As per claim 16, Donohue teaches the update service node of Claim 12, wherein the update module distributes software updates to client computers according to rules established by an administrator via the administration API using the administration user interface *(i.e. The structure of an updater component comprises data, methods for operating on that data, and a public application programming interface (API) which allows other updater components to contact and communicate with it. This structure will now be described in detail, col. 11, lines 21-25); and*

As per claim 17, Donohue teaches the update service node of Claim 11, wherein the child update service node may be selectively configured to periodically obtain available software updates from the parent update service node *(i.e. Updater components are preferably an integral part of the products they will serve to update. Hence, the updater component is distributed to*

software users together with an initial version of a software product, the updater component then automatically obtaining and applying software updates in accordance with preset criteria (such as a time period between successive searches for updates, and whether the particular user has selected to receive all updates or only certain updates--such as to receive updating patches but not replacement product versions for example), col. 5, lines 1-10).

Claim Rejections - 35 USC § 103

7. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

This application currently names joint inventors. In considering patentability of the claims under 35 U.S.C. 103(a), the examiner presumes that the subject matter of the various claims was commonly owned at the time any inventions covered therein were made absent any evidence to the contrary. Applicant is advised of the obligation under 37 CFR 1.56 to point out the inventor and invention dates of each claim that was not commonly owned at the time a later invention was made in order for the examiner to consider the applicability of 35 U.S.C. 103(c) and potential 35 U.S.C. 102(e), (f) or (g) prior art under 35 U.S.C. 103(a).

8. Claims 1-10, 25-29 are rejected under 35 U.S.C. 103(a) as being unpatentable over Donohue (US Patent No. 6,199,204), in view of Otto (US Patent No. 5,706,431).

As per claim 1, Donohue teaches a software update distribution system for distributing a software update over a communication network for distribution to client computers *(i.e. Thus, the present invention provides an agent and a method for obtaining and applying software updates which significantly reduces the cost and effort for software distributors of distributing and*

tracking software updates and significantly reduces the effort for system administrators and end users of applying updates to installed software, col. 5, lines 43-48), comprising:

a root update service node *(i.e. remote server system 50, col. 7, line 55 to col. 8, line 11; Fig. 3)*; and

a plurality of child update service nodes *(i.e. the updater component, col. 9, lines 51-67)* wherein the root update service node and the at least one child update service node are organized in a hierarchical manner such that the root update service node is a parent update service node to at least one child update service node *(See Fig. 3)*, wherein each update service node, except the root update service node, has a parent update service node *(i.e. remote server system 50, col. 7, line 55 to col. 8, line 11; Fig. 3)*, wherein each of the plurality of child update service nodes is configured to operate as a parent update service node to another child update service node *(i.e. repository 40', col. 7, lines 12-20; For an updater component to find and talk to another updater component on another remote machine the above information would have to be augmented by having a repository 40' which is accessible from both machines (preferably a central or distributed database accessible from anywhere in the network, such as a Web Page or pan-network file) and is available to all updater components that require it. Entries would be of the form updater_name machine_ip_address (OR DNS entry), port number, protocol, col. 7, lines 12-20); and*

wherein the root update service node *(i.e. the updater component, col. 9, lines 51-67)* obtains a software update from a software provider *(i.e. vendor, col. 7, line 55 to col. 8, line 11)*, and wherein each of the at least one child update service nodes obtains the software update for distribution to client *(i.e. user, col. 3, line 65 to col. 4, line 14)* computers by obtaining the

software update from its parent software update node *(i.e. the update component must obtain the required resources, col. 9, lines 51-67).*

Donohue does not specifically teach wherein at least one child update service node of the plurality of child update service nodes is a parent update service node to another child update service node of the plurality of child update service nodes.

Otto teaches wherein at least one child update service node of the plurality of child update service nodes is a parent update service node to another child update service node of the plurality of child update service nodes *(i.e. a first node of the communications network, for receiving the current status from the second node, determining as a function of the current status whether a revision of the second node information is required and, if the revision is required, transmitting the revision to the second node to revise the second node information and (3) second information revising circuitry, associated with the second node of the communications network, for receiving a current status from a third node of the communications network, determining as a function of the current status from the third node whether a revision of third node information stored in a memory of the third node is required and, if the revision is required, transmitting the revision received from the first node to the third node to revise the third node information, the revision thereby propagating through the communications network via the first, second and third nodes thereof, col. 2, lines 24-44).*

It would have been obvious to one of ordinary skill of the art having the teaching of Donohue and Otto at the time the invention was made to modify the system of Donohue to include the limitations as taught by Otto. One of ordinary skill in the art would be motivated to make this combination in order to transmit the revision received from the first node to the third

node to revise the third node information, the revision thereby propagating through the communications network via the first, second and third nodes thereof in view of Otto (col. 2, lines 24-44), as doing so would give the added benefit of allowing revisions to propagate automatically through a communications network as taught by Otto (col. 2, lines 45-58).

As to claims 25, 29, Donohue teaches a method for distributing software updates to a first child update service node from a second child update service node in a software update distribution system over a communication network, wherein the first and second child update services nodes are organized in a hierarchy of update service nodes originating with a root update service node *(i.e. Thus, the present invention provides an agent and a method for obtaining and applying software updates which significantly reduces the cost and effort for software distributors of distributing and tracking software updates and significantly reduces the effort for system administrators and end users of applying updates to installed software, col. 5, lines 43-48)*, the method comprising, as executed on the second child update service node:

(a) receiving a request for a software update catalog from the first child update service node, the software update catalog identifying software products for which the second child update service node provides software updates *(i.e. repository 40', col. 7, lines 12-20; For an updater component to find and talk to another updater component on another remote machine the above information would have to be augmented by having a repository 40' which is accessible from both machines (preferably a central or distributed database accessible from anywhere in the network, such as a Web Page or pan-network file) and is available to all*

updater components that require it. Entries would be of the form updater_name machine_ip_address (OR DNS entry), port number, protocol, col. 7, lines 12-20);

(b) returning a software update catalog to the first child update service node *(i.e. For example, the manufacturing department of an organisation may have three computer systems on which distributed software products collaborate with each other, the systems being called a, b and c. Typical entries in the Web page or file manufacturing_collaborators.html might be: ibm_catia_updater a.manufacturing.com 5000 tcp; ibm_db2pe_updater b.manufacturing.com 5100 tcp; ibm_cics_updater c.manufacturing.com 4780 tcp , col. 7, lines 22-29);*

(c) receiving a request for a list of software updates for a selected software product available for the first child update service node *(i.e. An updater component can then connect and talk to any other updater component using the DNS name to create an IP address and the port number which the remote updater component is listening to at that address, col. 7, lines 29-33);*

(d) determining whether any software updates for the selected software product are available for the first child update service node *(i.e. The updater component performs 290 (see FIGS. 3 and 4) a scan of the operating system file system to check whether the required software resources are already available on the local computer system, col. 9, lines 51-67);*

(e) returning a software update list identifying those software updates for the selected software product determined to be available for the first child update service node to the first child update service node, the software update list identifies available software updates according to a unique update identifier *(i.e. Since the updates list file 160 returned to the updater component in response to an initial search includes an identification 130 of pre-requisite software, that information enables the aforementioned examination 290 of the updater*

component registration database 40,40' to check whether pre-requisite software is available locally or remotely, col. 13, lines 22-41);

(f) receiving an update request for metadata corresponding to an available software update from the first child update service node, the update request identifying the available software update by its unique update identifier *(i.e. the updater component 20 of the first computer program contacts 300 this pre-requisite updater component 20' and requests that it attempt to update its associated pre-requisite software product 30'. This updater component 20' can, if necessary, request other updater components of its pre-requisite software to update their versions, and so on, col. 13, lines 22-41); and*

(g) returning update metadata corresponding to the software update identified in the update request, the update metadata including information corresponding to the software update including a reference for obtaining the corresponding update payload *(i.e. the updater component 20 of the first computer program contacts 300 this pre-requisite updater component 20' and requests that it attempt to update its associated pre-requisite software product 30'. This updater component 20' can, if necessary, request other updater components of its pre-requisite software to update their versions, and so on, col. 13, lines 22-41).*

Donohue does not explicitly teach wherein the second child update service node is a parent update service node to the first child update service node.

Otto teaches the second child update service node is a parent update service node to the first child update service node *(i.e. a first node of the communications network, for receiving the current status from the second node, determining as a function of the current status whether a revision of the second node information is required and, if the revision is required, transmitting*

the revision to the second node to revise the second node information and (3) second information revising circuitry, associated with the second node of the communications network, for receiving a current status from a third node of the communications network, determining as a function of the current status from the third node whether a revision of third node information stored in a memory of the third node is required and, if the revision is required, transmitting the revision received from the first node to the third node to revise the third node information, the revision thereby propagating through the communications network via the first, second and third nodes thereof, col. 2, lines 24-44).

It would have been obvious to one of ordinary skill of the art having the teaching of Donohue and Otto at the time the invention was made to modify the system of Donohue to include the limitations as taught by Otto. One of ordinary skill in the art would be motivated to make this combination in order to transmit the revision received from the first node to the third node to revise the third node information, the revision thereby propagating through the communications network via the first, second and third nodes thereof in view of Otto (col. 2, lines 24-44), as doing so would give the added benefit allowing revisions to propagate automatically through a communications network as taught by Otto (col. 2, lines 45-58).

As per claim 2, Donohue teaches the software update distribution system of Claim 1, wherein the root update service node comprises:

an update store for storing software updates *(i.e. remote server system 50, col. 7, line 55 to col. 8, line 11; Fig. 3);*

an update web service through which the root update service node distributes software updates to its child update service nodes over the communication network *(i.e. Each server system includes within storage a list 60 of the latest versions of, and patches for, software products which are available from that server. Each vendor is assumed here to make available via their Web sites such a list 60 of software updates (an example of which is shown in FIG. 2) comprising their product release history, in a format which is readable by updater components, and to make available the software resources 70 required to build the releases from a given level to a new level, col. 7, line 55 to col. 8, line 11); and*

a software provider interface through which a software provider submits its software update over the communication network to the root update distribution node *(i.e. Each server system includes within storage a list 60 of the latest versions of, and patches for, software products which are available from that server. Each vendor is assumed here to make available via their Web sites such a list 60 of software updates (an example of which is shown in FIG. 2) comprising their product release history, in a format which is readable by updater components, and to make available the software resources 70 required to build the releases from a given level to a new level, col. 7, line 55 to col. 8, line 11).*

As per claim 3, Donohue teaches the software update distribution system of Claim 2, wherein each of the plurality of child update service nodes comprises:

an update store for storing software updates *(i.e. remote server system 50, col. 7, line 55 to col. 8, line 11; Fig. 3);*

an update web service through which the child update service node obtains software updates from its parent update service node over the communication network, and through which the child update service node distributes software updates to its child update service nodes over the communication network *(i.e. Each server system includes within storage a list 60 of the latest versions of, and patches for, software products which are available from that server. Each vendor is assumed here to make available via their Web sites such a list 60 of software updates (an example of which is shown in FIG. 2) comprising their product release history, in a format which is readable by updater components, and to make available the software resources 70 required to build the releases from a given level to a new level, col. 7, line 55 to col. 8, line 11);*

an administration application programming interface (API) through which an administrator *(i.e. registration, col. 7, lines 34-45)*, using an administration user interface, establishes rules for distributing software updates to its child update service nodes *(i.e. The structure of an updater component comprises data, methods for operating on that data, and a public application programming interface (API) which allows other updater components to contact and communicate with it. This structure will now be described in detail, col. 11, lines 21-25); and*

a child update module for determining which software updates are available to be distributed to its child update service nodes according to the established rules *(i.e. repository 40', col. 7, lines 12-20; For an updater component to find and talk to another updater component on another remote machine the above information would have to be augmented by having a repository 40' which is accessible from both machines (preferably a central or distributed database accessible from anywhere in the network, such as a Web Page or pan-network file) and*

is available to all updater components that require it. Entries would be of the form updater_name machine_ip_address (OR DNS entry), port number, protocol, col. 7, lines 12-20).

As per claim 4, Donohue teaches the software update distribution system of Claim 3, wherein each of the plurality of child update service nodes further comprises:

a reporting module for generating and sending update activity reports to the parent update service node *(i.e. The updater generates 380 a report and writes 390 to log records, and then quits execution 400 (in the preferred embodiment the updater goes into a sleep or idle state) until activated again 410 upon expiry of a predetermined update cycle period (the repeat period parameter is configured when the updater component is installed), col. 11, lines 14-19);*

an authentication and authorization module for determining whether an update service mode is authorized to obtain software update from the child update service node *(i.e. The updater component preferably also includes a mechanism for verifying the authenticity of downloaded software, using cryptographic algorithms. This avoids the need for dedicated, password-protected or otherwise protected software resource repository sites. The software resources can be anywhere on the network as long as they are correctly named and or posted to the network search engines, col. 5, lines 36-42); and*

a client update module for distributing software updates to client computers *(i.e. user, col. 3, line 65 to col. 4, line 14).*

As per claim 5, Donohue teaches the software update distribution system of Claim 4, wherein the update store comprises an update content store in which the update payload for the

software update is stored, and an update information store in which update metadata for the software update is stored *(i.e. Product_ID; Current_Installed_Version; Current_License, col. 11, lines 26-39).*

As per claim 6, Donohue teaches the software update distribution system of Claim 5, wherein the child update service node obtains the software update from the parent update service by obtaining update metadata for the software update from the parent update service node, and separately obtaining the update payload for the software update from the parent update service node *(i.e. Since the updates list file 160 returned to the updater component in response to an initial search includes an identification 130 of pre-requisite software, that information enables the aforementioned examination 290 of the updater component registration database 40,40' to check whether pre-requisite software is available locally or remotely, col. 13, lines 22-41).*

As per claim 7, Donohue teaches the software update distribution system of Claim 6, wherein the child update service node obtains the update payload for the software update from the parent update service node in just-in-time fashion *(i.e. Each server system includes within storage a list 60 of the latest versions of, and patches for, software products which are available from that server. Each vendor is assumed here to make available via their Web sites such a list 60 of software updates (an example of which is shown in FIG. 2) comprising their product release history, in a format which is readable by updater components, and to make available the software resources 70 required to build the releases from a given level to a new level, col. 7, line 55 to col. 8, line 11).*

As per claim 8, Donohue teaches the software update distribution system of Claim 4, wherein the client update module distributes software updates to client computers according to rules established by an administrator via the administration API using the administration user interface *(i.e. The structure of an updater component comprises data, methods for operating on that data, and a public application programming interface (API) which allows other updater components to contact and communicate with it. This structure will now be described in detail, col. 11, lines 21-25); and*

As per claim 9, Donohue teaches the software update distribution system of Claim 8, wherein the root update service node further comprises a client update module for distributing software updates to client computers *(i.e. user, col. 3, line 65 to col. 4, line 14).*

As per claim 10, Donohue teaches the software update distribution system of Claim 3, wherein the child update service node may be selectively configured to periodically obtain available software updates from the parent update service node *(i.e. Updater components are preferably an integral part of the products they will serve to update. Hence, the updater component is distributed to software users together with an initial version of a software product, the updater component then automatically obtaining and applying software updates in accordance with preset criteria (such as a time period between successive searches for updates, and whether the particular user has selected to receive all updates or only certain updates--such as to receive updating patches but not replacement product versions for example), col. 5, lines 1-10).*

As per claim 26, Donohue teaches the method of claim 25 further comprising:

prior to steps (a)-(g):

receiving an authentication and authorization request from the first child update service node *(i.e. The structure of an updater component comprises data, methods for operating on that data, and a public application programming interface (API) which allows other updater components to contact and communicate with it. This structure will now be described in detail, col. 11, lines 21-25)*;

determining whether the first child update service node is authorized to obtain software updates from the second child update service node, *(i.e. The updater component includes the following persistent data: Product_ID: an identifier of the software product which is managed by this updater component. Current_Installed_Version: a version identifier for the installed software (e.g. version 3.1.0) Current_License: a version identifier corresponding to the software product version up to which the current software license allows the user to upgrade (e.g. version 4.0.z). Alternatively, this may be a licence identifier (e.g. LIC1) for use when accessing machine readable licence terms, col. 11, lines 26-39); and*

if the first child update service node is authorized to obtain software updates from the second child update service node, returning an authorization token to the first child update service node *(i.e. The updater component includes the following persistent data: Product_ID: an identifier of the software product which is managed by this updater component. Current_Installed_Version: a version identifier for the installed software (e.g. version 3.1.0) Current_License: a version identifier corresponding to the software product version up to which the current software license allows the user to upgrade (e.g. version 4.0.z). Alternatively, this*

may be a licence identifier (e.g. LIC1) for use when accessing machine readable licence terms, col. 11, lines 26-39); and

wherein each subsequent communication from the child update service node to the second child update service node is made with the authorization token *(i.e. The updater component includes the following persistent data: Product_ID: an identifier of the software product which is managed by this updater component. Current_Installed_Version: a version identifier for the installed software (e.g. version 3.1.0) Current_License: a version identifier corresponding to the software product version up to which the current software license allows the user to upgrade (e.g. version 4.0.z). Alternatively, this may be a licence identifier (e.g. LIC1) for use when accessing machine readable licence terms, col. 11, lines 26-39).*

As per claim 27, Donohue teaches the method of claim 26 further comprising:

receiving an update request for the update payload corresponding to the software update identified in the update request *(i.e. The structure of an updater component comprises data, methods for operating on that data, and a public application programming interface (API) which allows other updater components to contact and communicate with it. This structure will now be described in detail, col. 11, lines 21-25)*;

returning the update payload corresponding to the software update identified in the update request *(i.e. The updater component includes the following persistent data: Product_ID: an identifier of the software product which is managed by this updater component. Current_Installed_Version: a version identifier for the installed software (e.g. version 3.1.0) Current_License: a version identifier corresponding to the software product version up to which*

the current software license allows the user to upgrade (e.g. version 4.0.z). Alternatively, this may be a licence identifier (e.g. LIC1) for use when accessing machine readable licence terms, col. 11, lines 26-39).

As per claim 28, Donohue teaches the method of claim 25, wherein the request for a list of software updates for a selected software product includes an update anchor identifying the latest update information on the child update service node corresponding to the selected software product *(i.e. repository 40', col. 7, lines 12-20; For an updater component to find and talk to another updater component on another remote machine the above information would have to be augmented by having a repository 40' which is accessible from both machines (preferably a central or distributed database accessible from anywhere in the network, such as a Web Page or pan-network file) and is available to all updater components that require it. Entries would be of the form updater_name machine_ip_address (OR DNS entry), port number, protocol, col. 7, lines 12-20)*; *and*

wherein determining whether any software updates for the selected software product are available for the child update service node comprises determining whether there are any updates available to the child update service node according to the update anchor and according to rules associated with the distribution of software updates to the child update service node *(i.e. repository 40', col. 7, lines 12-20; For an updater component to find and talk to another updater component on another remote machine the above information would have to be augmented by having a repository 40' which is accessible from both machines (preferably a central or distributed database accessible from anywhere in the network, such as a Web Page or pan-*

network file) and is available to all updater components that require it. Entries would be of the form updater_name machine_ip_address (OR DNS entry), port number, protocol, col. 7, lines 12-20).

Response to Arguments

9. Applicant's arguments regarding Cheng does not teach the newly features of the amended claims 1, 11, 25, 29 have been considered but are moot in view of the new ground(s) of rejection.

Conclusion

10. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Miranda Le whose telephone number is (571) 272-4112. The examiner can normally be reached on Monday through Friday from 8:30 AM to 5:00 PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, John R. Cottingham, can be reached on (571) 272-7079. The fax number to this Art Unit is 571-273-8300.

Any inquiry of a general nature or relating to the status of this application should be directed to the Group receptionist whose telephone number is (703) 305-3900.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Miranda Le
September 14, 2007

JOHN COTTINGHAM
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100